TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

Assignment Title

BMCS2114 MACHINE LEARNING

2024/2025

Student's name/ ID Number	:	Ten Wei Kang / 22WMR14227
Student's name/ ID Number	:	Tan Ming Hue / 22WMR14324
Student's name/ ID Number	:	Chong Xu Ming / 22WMR14305
Student's name/ ID Number	:	- - -
Programme	:	RDS2
Tutorial Group	:	G4
Tutor's name	:	Dr Lim Siew Mooi

Table of Content

1.0 Project Description

1.1 Title

- Anomaly Detection in Network Traffic
- Dataset Link: https://www.kaggle.com/datasets/anushonkar/network-anamoly-detection?select=Network+Anamoly+Detection.docx

1.2 Abstract

Detecting anomalies in network traffic is essential for maintaining the security and reliability of modern networked systems. This research investigates the use of unsupervised learning, particularly clustering techniques, to identify and classify unusual activities in network traffic data. By utilizing evaluation metrics like the silhouette score and Davies-Bouldin score, we assess clustering quality and determine the optimal number of clusters. Our results show that clustering algorithms can effectively separate normal from abnormal network behaviors, enabling early detection of potential threats such as cyberattacks, irregular traffic patterns, and system malfunctions. The study also emphasizes the role of feature engineering and data preprocessing in improving the effectiveness of anomaly detection models. Through extensive experimentation and analysis, we demonstrate the practical application of clustering methods for real-time network surveillance and enhanced security.

1.3 Keywords

1. Anomaly Detection
   - Anomaly detection involves identifying data points, events, or behaviors that deviate from a system's normal patterns, often referred to as outliers or anomalies. In network traffic, this means detecting unusual activities or patterns that may signal security threats such as cyberattacks or operational problems. Real-time anomaly detection is vital for safeguarding the security and integrity of network systems, allowing administrators to address risks and prevent potential breaches.

2. Network Traffic
   - Network traffic refers to the data flows within a network, including packets of information transmitted between devices, encompassing all communication activities. Network traffic analysis involves monitoring and examining these flows to understand network behavior, tracking volume, frequency, and types of data, as well as sources and destinations. Understanding network traffic is essential for diagnosing issues, optimizing performance, and ensuring security. By analyzing traffic, administrators can identify patterns, detect anomalies, and take appropriate actions to maintain network health and security.

1.4 Introduction

In today's era , network traffic plays an important role for each and everyone of us in the modern world. This is because network traffic represents the transmission of information between devices in a computer network(What Is Network Traffic? Definition and How to Monitor It | Fortinet, n.d.) , and these devices not only included traditional devices like computers and

servers but also a vast array of interconnected systems such as routers , switches ,mobile devices and other networking equipment.

Web browsing, file transfer to video streaming and email exchanges are all examples of information transmission that can occur (IPCISCO, 2022). For example, when we stream a video on platforms like Netflix or YouTube, data is constantly being transmitted across the network to ensure that the content is delivered smoothly to our devices. Similarly, when businesses host virtual meetings or transfer large files between offices located in different regions, vast amounts of network traffic are generated to support these operations.

1.5 Problem statement

However , several factors have caused maintaining the network traffic more difficult. For example , due to the widespread use of cloud services , data-intensive apps and data intensive applications and linked devices has led to an increase of volume and complexity of data flowing through networks. This growing complexity makes it increasingly difficult to monitor and manage the network traffic effectively(What Are the Main Challenges and Limitations of Network Traffic Analysis for Infrastructure Security?, 2023).

Then in these days , our cybersecurity is difficult to keep up with these new and unidentified threats from the attackers that are using ever more sophisticated methods evolving cybersecurity threats to access our networks and steal data(Cisco: Over Half of Cyber Security Technologies in Malaysia Outdated – AMCHAM, n.d.). This has made abnormal detection become more and more important for network traffic.

1.6 Solution

So , we suspect that by giving our anomaly detection unsupervised learning algorithms the problem can be handled efficiently . This is because unsupervised learning is able to analyze or group datasets that are unlabeled by using machine learning techniques. For example by grouping the data into different clusters we can easily identify the data that may be a potential threat.

Furthermore ,unsupervised learning identifies hidden patterns in unlabeled data and having the ability to find patterns is valuable for detecting network abnormalities. For example , by grouping raw network traffic data based on similarities , allowing us to differentiate the difference between normal and suspicious activity of the traffic network.

Lastly, using unsupervised learning enables us to have more flexibility and scalability of ways to monitor the network traffic abnormal behaviors. This is because every cluster ensures that the flagged anomalies are meaningful and provide better protection for the increasingly complex network environment.

2.0 Literature Review

2.1 Autoencoder

Autoencoders are sophisticated neural networks employed in unsupervised learning, particularly for dimensionality reduction and anomaly detection. They comprise an encoder that compresses input data into a reduced-dimensional representation and a decoder that reconstructs the original data from this compressed form. Anomalies are detected by analyzing the reconstruction error, with data points exhibiting significant reconstruction errors being classified as anomalies.

Application:
Autoencoders have been successfully applied in various domains, including network intrusion detection (Ruff et al., 2019), fraud detection (Chen et al., 2019), and industrial anomaly detection (Zhou and Paffenroth, 2017).

2.2 GMM (Gaussian Mixture Model)

Gaussian Mixture Models (GMMs) are probabilistic models that assume data is generated from a combination of multiple Gaussian distributions. They are used for clustering and anomaly detection by pinpointing data points that poorly align with any of the Gaussian components.

Application:
GMMs have been applied in various fields, including image segmentation (Celebi et al., 2013), speech recognition (Reynolds, 2015), and anomaly detection in network traffic (Gao et al., 2014).

2.3 HDBSCAN ((Hierarchical Density-Based Spatial Clustering of Applications with Noise)

HDBSCAN, an extension of DBSCAN, uses hierarchical clustering based on density estimates. It excels at handling datasets with varying densities and can detect clusters of diverse shapes and sizes, with anomalies typically identified as points that fall outside any cluster.

Application:
HDBSCAN has been used in various applications, including image segmentation (Tung et al., 2018), bioinformatics (Moulton et al., 2019), and anomaly detection in sensor data (Breunig et al., 2000).

2.4 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN is a density-based clustering algorithm that groups together densely packed data points and designates those in sparse, low-density regions as outliers or anomalies. A key advantage of DBSCAN is that it does not necessitate the specification of the number of clusters in advance.

Application:
DBSCAN has been widely used in various fields, including geographic information systems (GIS) (Shekhar et al., 2003), network traffic analysis (Zhao et al., 2005), and anomaly detection in manufacturing (Kriegel et al., 2011).

2.5 Isolation Forest

Isolation Forest is an ensemble-based anomaly detection method that isolates anomalies by randomly partitioning the data. Anomalies are identified as points that are easier to isolate due to their rarity and distinctness.

Application:
Isolation Forest has been applied in various domains, including cybersecurity (Agarap, 2018), financial fraud detection (Aggarwal, 2015), and healthcare (Zhao et al., 2019).

2.0 Methodology

2.1 Data Source

The dataset used in this project is from Kaggle.com, which is a familiar website for machine learning students for learning various machine learning algorithms. Basically the dataset we used in the project is used to perform network abnormally detection and the dataset is called **train.txt** and then we combined the dataset into **networkDetect.csv**

BASIC FEATURES OF EACH NETWORK CONNECTION VECTOR

Attributes	**Description**
Duration	Length of time duration of the connection
Protocol_type	Protocol used in the connection
Service	Destination network service used
Flag	Status of the connection (Normal or Error)
Src_bytes	The total amount of data bytes transmitted from the source to the destination in a single connection.
Dst_bytes	The total amount of data bytes transmitted from the destination to the source in a single connection.
Land	If source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0
Wrong_fragment	The overall count of incorrect fragments in this connection.
Urgent	The number of urgent packets in this connection, where the urgent bit is activated.

CONTENT RELATED FEATURES OF EACH NETWORK CONNECTION VECTOR

Attributes	**Description**
Hot	Number of „hot“ indicators in the content such as: entering a system directory, creating programs and executing programs
Num_failed _logins	Count of failed login attempts
Logged_in Login Status	1 if successfully logged in; 0 otherwise
Num_compromised	Number of ``compromised' ' conditions
Root_shell	1 if root shell is obtained; 0 otherwise

Su_attempted	1 if ``su root'' command attempted or used; 0 otherwise
Num_root	Number of ``root'' accesses or number of operations performed as a root in the connection
Num_file_creations	Number of file creation operations in the connection
Num_shells	Number of shell prompts
Num_access_files	Number of operations on access control files
Num_outbound_cmds	Number of outbound commands in an ftp session
Is_hot_login	1 if the login belongs to the ``hot'' list i.e., root or admin; else 0
Is_guest_login	1 if the login is a ``guest'' login; 0 otherwise

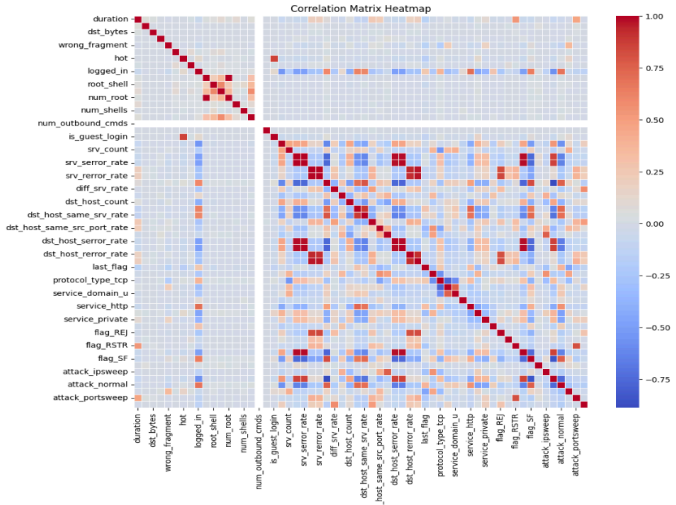
TIME RELATED TRAFFIC FEATURES OF EACH NETWORK CONNECTION VECTOR

Attributes	**Description**
Count	Number of connections to the same destination host as the current connection in the past two seconds
Srv_count	Number of connections to the same service (port number) as the current connection in the past two seconds
Serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)
Srv_serror_rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)
Rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)
Srv_rerror_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)
Same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in count (23)
Diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in count (23)
Srv_diff_host_ rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)

HOST BASED TRAFFIC FEATURES IN A NETWORK CONNECTION VECTOR

Attributes	**Description**
Dst_host_count	Number of connections having the same destination host IP address
Dst_host_srv_ count	Number of connections having the same port number
Dst_host_same _srv_rate	The percentage of connections that were to the same service, among the connections aggregated in dst_host_count (32)
Dst_host_diff_ srv_rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)
Dst_host_same_src_port _rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count (33)
Dst_host_srv_diff_host_r ate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count (33)
Dst_host_serro r_rate:	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32)
Dst_host_srv_s error_rate	The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count (33)
Dst_host_rerro r_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32)
Dst_host_srv_r error_rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_count (33)

2.2 Feature Selection



	feature_1	feature_2	correlation
717	num_root	num_compromised	0.998833
543	num_compromised	num_root	0.998833
1201	serror_rate	srv_serror_rate	0.993289
1259	srv_serror_rate	serror_rate	0.993289
1321	rerror_rate	srv_rerror_rate	0.989008
...	...	...	...
2399	protocol_type_udp	protocol_type_tcp	-0.771889
3063	flag_SF	attack_neptune	-0.845043
3237	attack_neptune	flag_SF	-0.845043
3210	attack_neptune	same_srv_rate	-0.887818
1470	same_srv_rate	attack_neptune	-0.887818

152 rows × 3 columns

By looking at the heatmap and the correlation of each data , the datasets that are highly correlalated features(above 0.9) , moderately correlated features(0.5-0.9) , diversity in metrics and multidimensional insight are mainly decided to use which are 'count', 'srv_count', 'dst_host_count', 'dst_host_srv_count','num_compromised', 'num_root', 'num_file_creations', 'num_shells', 'num_access_files', 'num_outbound_cmds', 'total_bytes', 'serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_same_srv_rate', 'dst_host_diff_srv_rate','dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate', 'dst_host_srv_serror_ rate', 'dst_host_rerror_rate' and 'dst_host_srv_rerror_rate' , then this ensures that the we capture a comprehensive view of the network activity , focusing on the traffic volume , error rates , system compromises , and service variety. Each group of features contributes to detecting network anomalies and identifying potential threats.

2.3 Sampling Method

2.3.1 Systematic Sampling

Systematic sampling is a probabilistic sampling technique wherein elements are selected from a larger population at regular, predefined intervals, referred to as the sampling interval. This approach guarantees that every member of the population has an equal probability of being chosen, though the selection process adheres to a systematic pattern.

The reason we perform systematic sampling for our dataset can be concluded from several aspect such as Execution time, memory usage and scalability.

- Execution Time: It has the fastest execution time across all sample fractions, even improving as the sample fraction increases. This makes it highly efficient for processing large datasets.
- Scalability: It scales exceptionally well, handling even the full dataset with virtually no increase in resource consumption

2.4 Model Selection

2.4.1 Autoencoder

Autoencoder is suitable for abnormal detection in network traffic due to its ability to learn a compressed, low-dimensional representation of data and reconstruct it. This is because autoencoders don't require label data , they learn patterns from the normal traffic and use that knowledge to detect anomalies . Furthermore , Autoencoders reduce the high-dimensional data into a compressed latent space and capture the most important patterns. Then, Autoencoder has the flexibility to adapt different types of data. Lastly, Autoencoders can scale efficiently to large datasets making it suitable for real-time network traffic analysis where vast amounts of data can be saved and analyzed.

2.4.2 Gaussian Mixture Model (GMM)

Gaussian Mixture Models (GMM) are effective for abnormal network detection because they can model complex data distributions and identify anomalous patterns flexibly and probabilistically. GMM represents data as a combination of several Gaussian distributions, assigning probabilities to each data point. This probabilistic approach is useful for detecting anomalies, as points with low probabilities (those that don't fit well with any of the Gaussian components) can be flagged as outliers. In network traffic, normal activities tend to cluster around common patterns, whereas abnormal behaviors often deviate significantly, resulting in lower probability values according to the GMM model.

2.4.3 HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is highly effective for abnormal network detection because it can identify clusters with varying densities and automatically detect outliers. In network traffic data, normal activities typically create dense clusters, while anomalies like unauthorized access or DDoS attacks often manifest as sparse, irregular points. HDBSCAN is adept at recognizing these diverse clusters and anomalies, even when they differ in density, which is essential for managing complex network environments.

Additionally, HDBSCAN's automatic outlier detection is valuable for anomaly detection, as unusual network activities often do not fit into well-defined clusters. By labeling these deviations as noise or outliers, HDBSCAN effectively isolates abnormal activities that would otherwise be overlooked by traditional clustering methods. This makes it particularly useful in identifying network attacks and irregular patterns.

2.4.4 DBSCAN

DBSCAN is well-suited for network traffic anomaly detection because of its capability to identify clusters of any shape and isolate outliers. Unlike methods that require a predefined number of clusters, DBSCAN operates without needing this specification and effectively manages noise in the dataset, which is common in network traffic due to packet loss and incomplete data. Its strengths in automatic outlier detection, handling arbitrary cluster shapes, and robustness to noise make DBSCAN particularly effective for detecting anomalies in network traffic.

2.4.5 Isolation Forest

Isolation Forest is highly suitable for abnormal network detection due to its ability to efficiently handle high-dimensional data and its focus on identifying outliers. Unlike traditional clustering techniques that aim to find dense clusters and label sparse points as outliers, Isolation Forest explicitly isolates anomalies by randomly partitioning the feature space. Since anomalous behavior in network traffic, such as unauthorized access or distributed denial-of-service (DDoS) attacks, is often rare and distinct, Isolation Forest isolates these outliers quickly and accurately.

In addition, network traffic data often involves a large number of features, such as packet size, duration, and source/destination IP addresses. Isolation Forest performs well in such high-dimensional spaces because it does not rely on computationally expensive distance-based metrics. This makes it ideal for detecting complex, multi-feature network anomalies that may arise from a wide range of potential attack vectors.

2.5 Hyperparameter Tuning

2.5.1 Data Tuning

Data tuning refers to the process of optimizing and refining data for better performance in machine learning models or other data-driven tasks. It involves adjusting various aspects of the data to improve the accuracy, efficiency, and overall outcomes of a model or analysis. This can include modifying the data's format, structure, scale, or the way it interacts with algorithms.

```
columns_to_standardize = [
    'duration', 'count', 'srv_count', 'dst_host_count', 'dst_host_srv_count',
    'num_compromised', 'num_root', 'num_file_creations', 'num_shells',
    'num_access_files', 'num_outbound_cmds', 'total_bytes',
    'serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
    'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_same_srv_rate',
    'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
    'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
    'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
    'dst_host_srv_rerror_rate'
]
```

Above image shows the column that is suitable to perform the scaling method.

2.5.1.1 Standard Scaling

```
# Apply StandardScaler
scaler_standard = StandardScaler()
data_standard_scaled = scaler_standard.fit_transform(data_to_standardize)
data_standard_scaled_df = pd.DataFrame(data_standard_scaled, columns=columns_to_standardize)
```

Standard Scaling is ideal for our dataset as it normalizes each feature to have a mean of zero and a standard deviation of one. This adjustment ensures that all features are on an equivalent scale, thereby mitigating the risk of any single feature with a larger magnitude exerting undue influence on the learning process.

Purpose of using Standard Scaling:

Enhances Model Performance: StandardScaler optimizes the performance and convergence of machine learning models, particularly those that are sensitive to feature scaling, such as linear regression, logistic regression, and support vector machines.

Facilitates Interpretability:StandardScaler simplifies the interpretation of coefficients or weights in linear models by standardizing the features, making it easier to understand their contributions.

Robust to Outliers: In contrast to min-max scaling, StandardScaler exhibits greater resilience to outliers, as it relies on the mean and standard deviation rather than the range of the data.

2.5.1.2 Min Max Scaling

```
# Apply MinMaxScaler
scaler_minmax = MinMaxScaler()
data_minmax_scaled = scaler_minmax.fit_transform(data_to_standardize)
data_minmax_scaled_df = pd.DataFrame(data_minmax_scaled, columns=columns_to_standardize)
```

Min-Max Scaling, also known as normalization, is a data preprocessing method that transforms features to a designated range, typically between 0 and 1. This technique is commonly used in machine learning to standardize the range of independent variables or features.

Why Min-Max scaling is useful:

- **Algorithm Performance**: Some machine learning algorithms (like neural networks and gradient-based methods) perform better or converge faster when the input features are scaled. Features with vastly different ranges can cause algorithms to struggle with convergence.

- **Equal Weighting**: Min-Max Scaling standardizes features to a consistent scale, ensuring that each feature contributes equally to the model.

2.5.2 Model Tuning

```
Best Silhouette Score for MinMax Scaled Data: 0.5916
Best Parameters for MinMax Scaled Data: {'n_components': 3, 'covariance_type': 'tied'}
Best Silhouette Score for Standard Scaled Data: 0.4644
Best Parameters for Standard Scaled Data: {'n_components': 3, 'covariance_type': 'tied'}
```

Using model tuning to focus on optimizing the parameters and structure of the machine learning model itself, the goal is to improve the model's performance by adjusting hyperparameters. This process includes techniques like hyperparameter tuning, model selection, regularization, and cross-validation to ensure the model generalizes well to new data. In our case, based on the diagram, we employed the silhouette score to evaluate the effectiveness of our model tuning. We discovered that after applying Min-Max Scaling and Standard Scaling to our dataset, the silhouette scores achieved were 0.5916 and 0.4644, respectively. These results indicate that the Min-Max Scaled data led to better clustering performance, providing clearer separation between the clusters compared to the Standard Scaled data. Thus, through both model and data tuning, we have identified the optimal data transformation that enhances our model’s ability to distinguish between clusters.

2.6 Evaluation

2.6.1 Visualization

In machine learning, visualization methods such as PCA (Principal Component Analysis), UMAP (Uniform Manifold Approximation and Projection), and t-SNE (t-distributed Stochastic Neighbor Embedding) are used for dimensionality reduction and data visualization. These techniques allow high-dimensional data to be projected into lower-dimensional spaces, usually 2D or 3D, making it easier to visualize patterns, clusters, and structures within the data.

2.6.1.1 TSNE (t-distributed Stochastic Neighbor Embedding)

t-SNE (t-distributed Stochastic Neighbor Embedding) is a non-linear dimensionality reduction technique designed for visualizing high-dimensional data. It excels at preserving the local structure of data by maintaining the pairwise distances between points, making it particularly effective for revealing patterns or clusters that are difficult to detect in higher dimensions. t-SNE works by converting Euclidean distances between points into probabilities representing similarities and then minimizes the divergence between these probability distributions in both the high- and low-dimensional spaces. This process results in a visual representation where similar points are clustered together, revealing hidden relationships within the data.

t-SNE is commonly used in applications like clustering, image recognition, and exploring complex data structures. Its strength lies in its ability to uncover non-linear relationships, often producing clear and visually distinct clusters. However, it can be computationally expensive, sensitive to hyperparameters (such as perplexity), and is primarily a tool for visualization rather than general dimensionality reduction. An example use case would be visualizing how samples from different classes in a dataset are distributed in a 2D or 3D space.

2.6.1.2 UMAP (Uniform Manifold Approximation and Projection)

UMAP (Uniform Manifold Approximation and Projection) is a non-linear dimensionality reduction method that excels in both visualization and general reduction, especially for high-dimensional datasets. It constructs a graph from the high-dimensional data and optimizes its representation in a lower-dimensional space. UMAP is adept at preserving both local relationships and global structure, making it highly effective for large datasets, such as those involving text, images, and genomic information.

UMAP is often preferred for its speed and scalability compared to other methods like t-SNE, while also providing more interpretable results. It efficiently handles large datasets and offers flexibility in balancing local and global structure through its hyperparameters, like n_neighbors. However, its performance can vary depending on these parameters, and it may not be ideal for very small datasets. A typical use case is visualizing embeddings from high-dimensional data such as image or text datasets, where capturing intricate patterns and maintaining structural relationships is essential.

2.6.1.3 PCA (Principal Component Analysis)

PCA (Principal Component Analysis) is a linear technique for dimensionality reduction that reorients data into a new coordinate system, where the axes, known as principal components, represent directions of

maximum variance. It identifies principal components that capture the most variation in the data, with the first component capturing the highest variance, followed by subsequent components. PCA simplifies datasets by reducing their dimensionality while maintaining as much of the original variance as possible, aiding in visualization and enhancing model performance.

PCA is often employed when the data is linearly separable and when there is a need to simplify high-dimensional data. Its strengths lie in its speed, ease of interpretation, and ability to work well with linearly structured data. However, it may struggle with capturing non-linear relationships. A common example is reducing a high-dimensional dataset into 2D for visualization purposes, such as using a scatter plot to identify trends or clusters.

2.6.2 Silhouette Score

The silhouette score is an important metric for assessing clustering quality, evaluating how closely an entity matches its own cluster (cohesion) compared to other clusters (separation). The score ranges from -1 to 1, where values near 1 suggest well-defined clusters, while values close to -1 indicate potential misclassification. A score of zero implies that a data point is near the boundary between two clusters, suggesting possible ambiguities. This metric is especially useful for determining the optimal number of clusters (k) and for comparing different clustering algorithms. By computing the silhouette score for various k values, one can identify the configuration that achieves the highest score, indicating the most effective clustering arrangement.

The silhouette score provides insights into the cohesion and separation of clusters, helping to assess the quality of clustering results. High cohesion implies that data points within a cluster are closely related, while high separation indicates distinct clusters. This makes the silhouette score a robust tool for applications like network anomaly detection, where a higher score indicates better differentiation between normal and abnormal activities. However, the silhouette score has limitations, such as not performing well on datasets with overlapping clusters or non-globular shapes, and being sensitive to data scale. Standardizing the data before computing the silhouette score can mitigate these issues.

In summary, the silhouette score is a powerful metric for evaluating clustering quality, aiding in the selection of the optimal number of clusters and the comparison of clustering algorithms. By leveraging the silhouette score, practitioners can enhance the accuracy and reliability of their clustering results, particularly in applications like network anomaly detection. Despite its limitations, the silhouette score remains a crucial tool for ensuring the appropriateness and effectiveness of clustering outcomes.

2.6.3 Davies-Bouldin Score

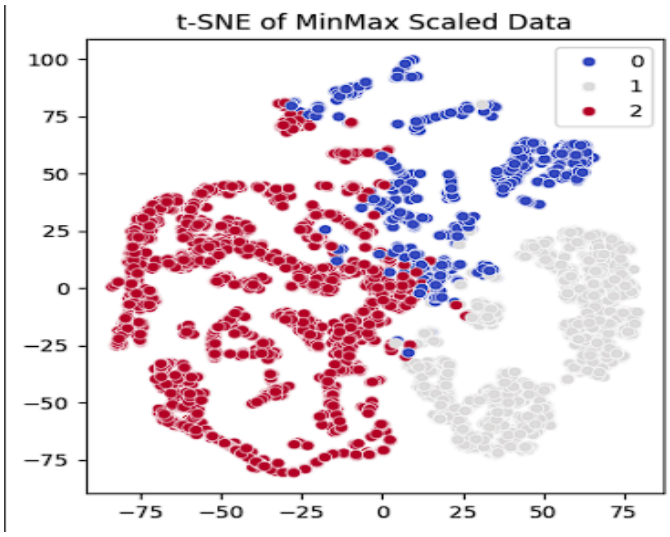
The Davies-Bouldin score is a metric for assessing clustering quality by calculating the average similarity ratio between each cluster and its most similar neighboring cluster. Lower scores indicate better clustering quality. This score is based on two main components: within-cluster dispersion and between-cluster separation. Within-cluster dispersion measures how closely data points are grouped within each cluster, while between-cluster separation evaluates the distance between different clusters. The Davies-Bouldin score combines both cluster compactness and separation to provide a thorough measure of clustering effectiveness. A lower score signifies that clusters are more compact and well-separated, reflecting a more successful clustering outcome.

The Davies-Bouldin score is useful for comparing various clustering algorithms and determining the optimal number of clusters (k) in a dataset. By evaluating the score across different k values, one can identify the clustering configuration that minimizes the score, indicating the most effective clustering arrangement. This makes the Davies-Bouldin score a valuable tool for evaluating clustering results and applications like network anomaly detection, where a lower score suggests better differentiation between normal and abnormal activities. However, it has limitations, such as struggling with overlapping clusters or non-globular shapes and being sensitive to data scale. Standardizing the data before calculating the Davies-Bouldin score can help address these issues.

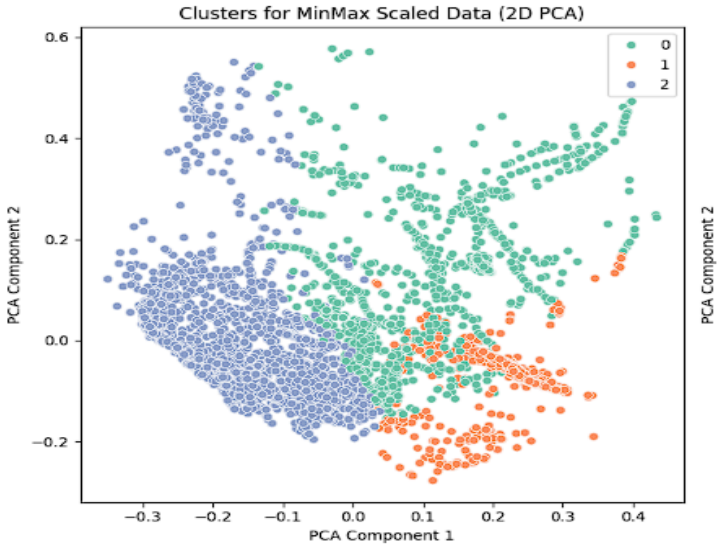
In summary, the Davies-Bouldin score is a robust metric for evaluating the quality of clustering in a dataset. It provides insights into the compactness and separation of clusters, helping to identify the optimal number of clusters and assess the performance of clustering algorithms. By leveraging the Davies-Bouldin score, practitioners can enhance the accuracy and reliability of their clustering results, particularly in applications like network anomaly detection. Despite its limitations, the Davies-Bouldin score remains a crucial tool for ensuring the appropriateness and effectiveness of clustering outcomes.

3.0 Result

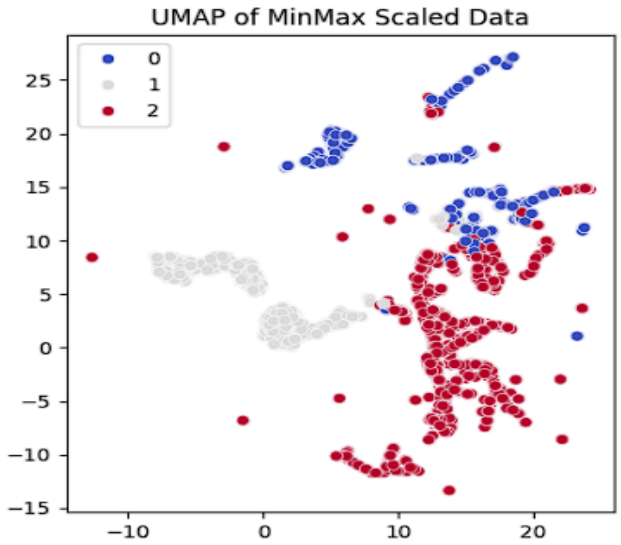
3.1 Min-Max Scaler Dataset



The image displays a t-SNE plot of data that has been MinMax scaled. The clustering was performed with n_clusters=3. The silhouette score indicates how well the clusters are separated and how consistent the points are within each cluster..



The image displays the result of clustering using PCA. The result shows a crucial message that Clusters are better distributed across the principal component space in the MinMax scaled data. All three clusters (blue, green, and orange) are well-separated with a more balanced spread, indicating that MinMax scaling preserves the relative distances between points while ensuring data remains within a specific range (likely [0, 1]). This suggests that MinMax scaling is effective when data should be normalized within a defined range, providing a clear separation of clusters.



The clusters (blue, gray, and red) in the MinMax scaled data appear more intermixed, with regions where clusters overlap, particularly between the red and blue points. There is no clear separation between the three clusters in this case. The points are spread out across the UMAP space, but with less distinct boundaries between clusters. This suggests that while MinMax scaling allows for more variation across the axes, it does not create as distinct a separation for UMAP as it did with PCA. Clusters 1 (gray) and 2 (red) are relatively mixed, while cluster 0 (blue) shows some separation but still overlaps with others.

```
# Apply K-Means on MinMax Encoded Data
kmeans_minmax = KMeans(n_clusters=3, random_state=42).fit(minmax_encoded)
silhouette_minmax = silhouette_score(minmax_encoded, kmeans_minmax.labels_)
```

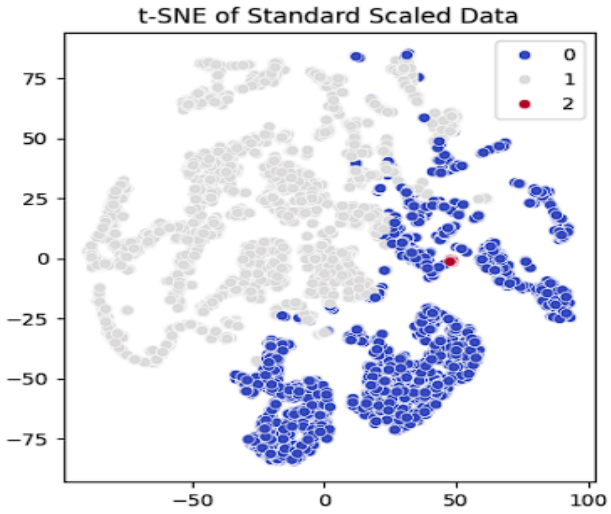
```
Silhouette Score for MinMax Scaled Data: 0.45855963230133057
```

The silhouette score of 0.4585 for the autoencoder reflects a moderate level of clustering effectiveness. This score indicates that the autoencoder has managed to achieve a reasonable degree of separation between clusters while maintaining good cohesion within each cluster. Data points within the same cluster are relatively close to one another, which suggests that the autoencoder has effectively captured important features and patterns in the data. However, the score being below 0.5 also indicates that the separation between clusters is not entirely distinct, with some overlap or boundary ambiguity between the clusters. This shows that while the autoencoder performs adequately, there is still potential for improvement.

The score highlights the autoencoder's strengths in dimensionality reduction, showing that it has been able to compress the data and reveal underlying structures. However, achieving optimal clustering remains a challenge, as the autoencoder may not have fully captured the ideal feature set or relationships between data points needed to completely separate the clusters. Further improvements could be pursued by refining the autoencoder's architecture, exploring different activation functions, layers, or even tweaking the number of latent dimensions.

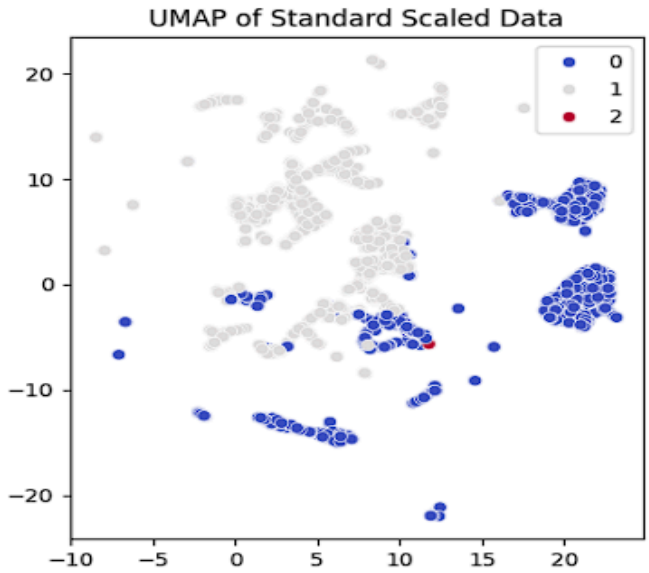
In addition to architectural adjustments, tuning hyperparameters such as learning rates, batch sizes, or epochs could further optimize the autoencoder's performance. Moreover, the choice of clustering algorithm also plays a significant role in the final output. Testing different clustering methods like DBSCAN, Gaussian Mixture Models, or fine-tuning the parameters of K-means could lead to clearer cluster boundaries and improved results. These adjustments could enhance both the autoencoder's ability to extract meaningful features and the clustering model's ability to form distinct, well-separated groups, potentially leading to a higher silhouette score and better overall performance.

3.2 Standard Scaler Dataset



The image shows a t-SNE plot of data scaled using a Standard Scaler, The red cluster appears either very small or overlapped by the other clusters, which suggests poor separation or an imbalance in cluster size. The red cluster's disappearance may point to suboptimal clustering, where the boundaries between clusters are unclear.

In contrast, one of the clusters (blue) is significantly stretched across the principal component space in the Standard scaled data, while the other two clusters (green and orange) are more compact and grouped together. The large spread of one cluster along the first PCA component suggests that Standard scaling, which standardizes features by removing the mean and scaling to unit variance, has caused higher variance in the data, impacting the clustering structure. Standard scaling may not have worked as well here, as the large spread of the data makes interpretation of clusters more difficult.



The UMAP plot of Standard Scaled data shows better cluster separation compared to the MinMax scaled plot, with more distinct and well-defined clusters. Clusters, such as cluster 0 (blue), cluster 1 (gray), and cluster 2 (red), are more clearly separated, with tighter and less dispersed groupings, indicating improved compactness. Additionally, fewer outliers are present, suggesting that Standard scaling may have enhanced the clustering of similar points and resulted in a more cohesive representation of the data.

```
# Apply K-Means on Standard Encoded Data
kmeans_standard = KMeans(n_clusters=3, random_state=42).fit(standard_encoded)
silhouette_standard = silhouette_score(standard_encoded, kmeans_standard.labels_)
```

```
Silhouette Score for Standard Scaled Data: 0.3897148668766022
```

The silhouette score of 0.3897 for the autoencoder on the standard scaler dataset indicates that the clustering results are moderately effective. This score suggests that the autoencoder has done a decent job of grouping similar data points together, but there is still some overlap between the clusters. A silhouette score in this range implies that the clusters are reasonably well-separated, which is a positive outcome, but it also highlights areas where improvements can be made. For instance, the autoencoder might benefit from further tuning, such as adjusting the architecture, increasing the number of epochs, or experimenting with different preprocessing techniques.

Moreover, the context of the problem plays a crucial role in interpreting this score. In some applications, a silhouette score of 0.3897 might be considered acceptable, especially if the primary goal is to achieve a general grouping of similar data points. However, for more critical applications where precise and distinct clusters are necessary, this score might indicate the need for more rigorous optimization. It is essential to consider the specific requirements and constraints of the task at hand when evaluating the performance of the autoencoder.

In conclusion, while the silhouette score of 0.3897 is not indicative of poor performance, it does suggest that there is room for improvement. This score provides a solid foundation for further refinement and optimization of the autoencoder's architecture and training parameters. By exploring additional techniques and fine-tuning the model, it is possible to enhance the clustering quality and achieve more distinct and well-separated clusters. Ultimately, the goal should be to maximize the silhouette score to ensure that the autoencoder delivers the best possible clustering results for the given dataset.

4.0 Conclusion

All in all , by comparing all five models that we are using , we found out that the best performing model for abnormal network detection is Auto Encoder , as it has the best clustering algorithm result compared to all other models. This is because that autoencoder has the ability to learn and compress the normal patterns of network activity into a lower-dimensional space through an unsupervised learning process. By training exclusively on data representing normal behavior, autoencoders develop a nuanced understanding of what constitutes typical network activity.

However Autoencoder is still having some disadvantages and limitations Such deviations might include unusual traffic patterns or signs of security breaches, which are not well-reconstructed by the autoencoder, signaling potential issues.

But for future improvement , we can use the flexibility of autoencoders to incorporate deep learning architectures that allow them to handle the vast and varied data types found in network environments, further enhancing their detection capabilities. This is because the ability of autoencoders to automatically extract meaningful features and detect outliers through reconstruction errors makes them superior for maintaining robust network security in a wide range of situations. This makes autoencoders not just a viable choice, but often the best one for anomaly detection in complex network environments.

5.0 References

1. What is network traffic? definition and how to monitor it | Fortinet. (n.d.). Fortinet. https://www.fortinet.com/resources/cyberglossary/network-traffic
2. IPCISCO. (2022, January 19). Network Traffic Types ⋆ IPCiSCO. IPCisco. https://ipcisco.com/lesson/network-traffic-types/
3. What are the main challenges and limitations of network traffic analysis for infrastructure security? (2023, September 1). https://www.linkedin.com/advice/0/what-main-challenges-limitations-network
4. What are the main challenges and limitations of network traffic analysis for infrastructure security? (2023, September 1). https://www.linkedin.com/advice/0/what-main-challenges-limitations-network
5. GeeksforGeeks. (2024, March 11). K means Clustering Introduction. GeeksforGeeks. https://www.geeksforgeeks.org/k-means-clustering-introduction/
6. Yenigün, O. (2024, March 11). DBSCAN Clustering Algorithm demystified. Built In. https://builtin.com/articles/dbscan#:~:text=What%20Is%20DBSCAN%3F-,Density%2Dbased%20spatial%20clustering%20of%20applications%20with%20noise%20(DBSCAN),data%20cleaning%20and%20outlier%20detection.
7. Sartorius. (n.d.). What is Principal Component Analysis (PCA) and how it is used? Sartorius. https://www.sartorius.com/en/knowledge/science-snippets/what-is-principal-component-analysis-pca-and-how-it-is-used-507186#:~:text=The%20goal%20is%20to%20extract,in%20the%20least%20squares%20sense.
8. GeeksforGeeks. (2024b, July 15). What is Isolation Forest? GeeksforGeeks. https://www.geeksforgeeks.org/what-is-isolation-forest/a
9. GeeksforGeeks. (2023b, December 6). Autoencoders -Machine learning. GeeksforGeeks. https://www.geeksforgeeks.org/auto-encoders/
10. Solutions, V. T. (2024, April 3). K-Means Clustering: applications and real-world use cases. https://www.linkedin.com/pulse/k-means-clustering-applications-real-world-mz4df/
11. DBSCAN | Engati. (n.d.). Engati. https://www.engati.com/glossary/dbscan#:~:text=%E2%80%8D%E2%80%8D-,What%20is%20DBSCAN%20used%20for%3F,based%20on%20the%20purchasing%20patterns.
12. What are the real-world applications of Principal Component Analysis? (2023, October 26). www.linkedin.com. https://www.linkedin.com/advice/0/what-real-world-applications-principal-component-analysis-crhxc
13. Kumar, S. (2022, January 4). 7 Applications of Auto-Encoders every Data Scientist should know. Medium. https://towardsdatascience.com/6-applications-of-auto-encoders-every-data-scientist-should-know-dc703cbc892b
14. Network Traffic Anomaly Detection with Machine Learning. (2024, June 17). Eyer. https://eyer.ai/blog/network-traffic-anomaly-detection-with-machine-learning/
15. GeeksforGeeks. (2024, February 9). What is StandardScaler? GeeksforGeeks. https://www.geeksforgeeks.org/what-is-standardscaler/
16. StandardScaler. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

17. MinMaxScaler. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
18. GeeksforGeeks. (2021, September 13). Systematic sampling in pandas. GeeksforGeeks. https://www.geeksforgeeks.org/systematic-sampling-in-pandas/
19. Pius, A. (2023, April 7). Introduction to all types of Autoencoders with Python. Medium. https://medium.com/chat-gpt-now-writes-all-my-articles/introduction-to-all-types-of-autoencoders-with-python-ec0e47b5e1b9
20. IsolationForest. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html
21. silhouette_score. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
22. davies_bouldin_score. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html
23. 2.3. Clustering. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/clustering.html
24. GeeksforGeeks. (2023, March 16). Exploring correlation in Python. GeeksforGeeks. https://www.geeksforgeeks.org/exploring-correlation-in-python/